Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software

Congestion management controls congestion after it has occurred on a network. Congestion can be managed on Cisco IOS XR software by using packet queueing methods, and by shaping the packet flow through use of traffic regulation mechanisms.

Packet queueing methods define packet scheduling or the order in which packets are dequeued to the interface for transmission on the physical wire. Furthermore, queueing methods support minimum bandwidth guarantees and low latencies based on the order and number of times that a queue is serviced.

The following types of queueing methods and traffic regulation mechanisms are supported on the Cisco IOS XR software:

- Modified Deficit Round Robin (MDRR)
- Low-latency queueing (LLQ) with strict priority queueing (PQ)
- Traffic shaping
- Traffic policing

This module provides the conceptual and configuration information for Quality of Service (QoS) packet queueing methods and traffic regulation mechanisms on Cisco IOS XR software.

Note For additional conceptual information about QoS in general and complete descriptions of the QoS commands listed in this module, see the "Related Documents" section of this module. To locate documentation for other commands that might appear in the course of executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Configuring Modular QoS Congestion Management on Cisco IOS XR Software Contents

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	This feature was first supported on the Cisco XR 12000 Series Router with the exception of the following: Two-rate policer and two-token bucket algorithm. The **pir** and **violate-action** keywords for the **police** command. Packet-by-packet MDRR scheduling mechanism. Added configuration examples.

Contents

Prerequisites for Configuring QoS Congestion Management on Cisco IOS XR Software

The following prerequisites are required for configuring QoS congestion management on your network:

- You must be in a user group associated with a task group that includes the proper task IDs for QoS commands. Task IDs for commands are listed in the *Cisco IOS XR Task ID Reference Guide*.

  For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- You must be familiar with Cisco IOS QoS configuration tasks and concepts.

Information About Configuring QoS Congestion Management on Cisco IOS XR Software

To implement QoS congestion management features in this document, you must understand the following concepts:

Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which a traffic flow (or packets) is sent out an interface based on priorities assigned to packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission. The congestion management features in Cisco IOS XR software allow you to specify creation of a different number of queues, affording greater or lesser degree of differentiation of traffic, and to specify the order in which that traffic is sent.

During periods with light traffic flow, that is, when no congestion exists, packets are sent out the interface as soon as they arrive. During periods of transmit congestion at the outgoing interface, packets arrive faster than the interface can send them. If you use congestion management features, packets accumulating at an interface are queued until the interface is free to send them; they are then scheduled for transmission according to their assigned priority and the queueing method configured for the interface. The router determines the order of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

In addition to queueing methods, QoS congestion management mechanisms, such as policers and shapers, are needed to ensure that a packet adheres to a contract and service. Both policing and shaping mechanisms use the traffic descriptor for a packet. See the "Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software" module for information about the traffic descriptor.

Policers and shapers usually identify traffic descriptor violations in an identical manner through the token bucket mechanism, but they differ in the way they respond to violations. A policer typically drops traffic flow; whereas, a shaper delays excess traffic flow using a buffer, or queueing mechanism, to hold Low-Latency Queueing feature that uses a strict priority queue with policing to shape the flow.)

Traffic shaping and policing can work in tandem. For example, a good traffic shaping scheme should make it easy for nodes inside the network to detect abnormal flows. This activity is sometimes called policing the traffic of the flow.

Modified Deficit Round Robin

MDRR is a class-based composite scheduling mechanism that allows for queueing of up to eight traffic classes. It operates in the same manner as class-based weighted fair queueing (CBWFQ) and allows definition of traffic classes based on customer match criteria (such as access lists); however, MDRR does not use the weighted fair queueing algorithm.

With MDRR configured in the queueing strategy, nonempty queues are served one after the other. Each time a queue is served, a fixed amount of data is dequeued. The algorithm then services the next queue. When a queue is served, MDDR keeps track of the number of bytes of data that were dequeued in excess of the configured value. In the next pass, when the queue is served again, less data is dequeued to

compensate for the excess data that was served previously. As a result, the average amount of data dequeued per queue is close to the configured value. In addition, MDRR allows for a strict priority queue for delay-sensitive traffic.

Each queue within MDRR is defined by two variables:

- Quantum value—Average number of bytes served in each round.
- Deficit counter—Number of bytes a queue has sent in each round. The counter is initialized to the quantum value.

Packets in a queue are served as long as the deficit counter is greater than zero. Each packet served decreases the deficit counter by a value equal to its length in bytes. A queue can no longer be served after the deficit counter becomes zero or negative. In each new round, the deficit counter for each nonempty queue is incremented by its quantum value.

Note In general, the quantum size for a queue should not be smaller than the maximum transmission unit (MTU) of the interface to ensure that the scheduler always serves at least one packet from each nonempty queue.

The Cisco CRS-1 implements a slight variation of the MDRR scheduling mechanism called packet-by-packet MDRR (P2MDRR). Using P2MDRR, queues are scheduled after every packet is sent compared to MDRR in which queues are scheduled after a queue is emptied. All non-high-priority queues with minimum bandwidth guarantees use P2MDRR.

Low-Latency Queueing with Strict Priority Queueing

The LLQ feature brings strict PQ to the MDRR scheduling mechanism. PQ in strict priority mode ensures that one type of traffic is sent, possibly at the expense of all others. For PQ, a low-priority queue can be detrimentally affected, and, in the worst case, never allowed to send its packets if a limited amount of bandwidth is available or the transmission rate of critical traffic is high.

Strict PQ allows delay-sensitive data, such as voice, to be dequeued and sent before packets in other queues are dequeued.

LLQ enables use of a single, strict priority queue within MDRR at the class level, allowing you to direct traffic belonging to a class. To rank class traffic to the strict priority queue, you specify the named class within a policy map and then configure the **priority** command for the class. (Classes to which the **priority** command is applied are considered priority classes.) Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

Through use of the **priority** command, you can assign a strict PQ to any of the valid match criteria used to specify traffic. These methods of specifying traffic for a class include matching on access lists, protocols, IP precedence, and IP differentiated service code point (DSCP) values. Moreover, within an access list you can specify that traffic matches are allowed based on the DSCP value that is set using the first six bits of the IP type of service (ToS) byte in the IP header.

Traffic Shaping

Traffic shaping allows you to control the traffic flow exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it. Traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

To match the rate of transmission of data from the source to the target interface, you can limit the transfer of data to one of the following:

- A specific configured rate
- A derived rate based on the level of congestion

The rate of transfer depends on these three components that constitute the token bucket: burst size, mean rate, and time (measurement) interval. The mean rate is equal to the burst size divided by the interval.

When traffic shaping is enabled, the bit rate of the interface does not exceed the mean rate over any integral multiple of the interval. In other words, during every interval, a maximum of burst size can be sent. Within the interval, however, the bit rate may be faster than the mean rate at any given time.

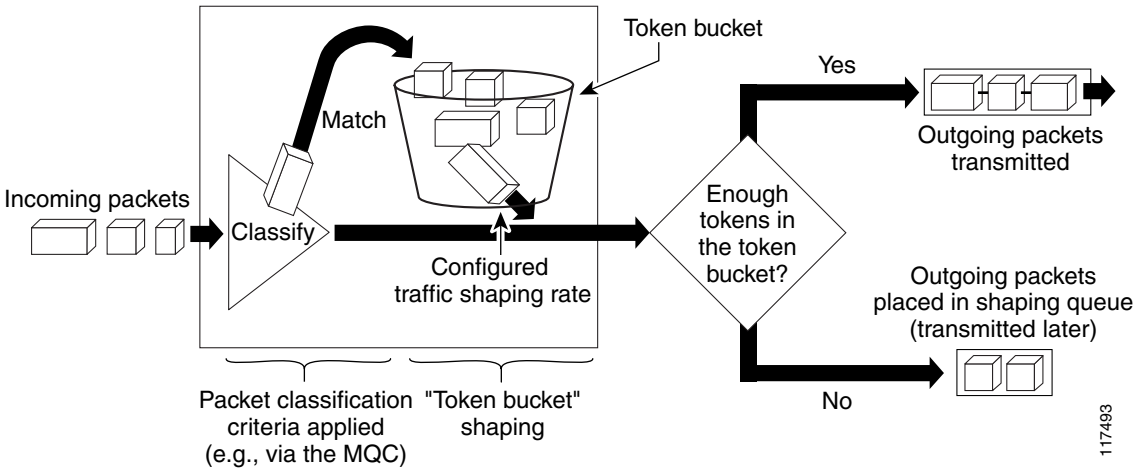
When the exceed burst size (Be) equals 0, the interface sends no more than the burst size every interval, achieving an average rate no higher than the mean rate. However, when the Be size is greater than 0, the interface can send as many as the conform burst size (Bc) plus Be bits in a burst, if in a previous time period the maximum amount was not sent. Whenever less than the burst size is sent during an interval, the remaining number of bits, up to the Be size, can be used to send more than the burst size in a later interval.

Traffic Shaping Mechanism Regulates Traffic

When incoming packets arrive at an interface. The packets are classified using a classification technique, such as use of an access control list (ACL) or setting of the IP Precedence bits through the Modular QoS CLI (MQC). If the packet matches the specified classification, the traffic shaping mechanism continues. Otherwise, no further action is taken.

Figure 2 illustrates how a traffic shaping mechanism regulates traffic flow.

Figure 2 How a Traffic Shaping Mechanism Regulates Traffic



Packets matching the specified criteria are placed in the token bucket. The maximum size of the token bucket is the confirm burst (Bc) size plus the Be size. The token bucket is filled at a constant rate of Bc worth of tokens at every Tc. This is the configured traffic shaping rate.

If the traffic shaping mechanism is active (that is, packets exceeding the configured traffic shaping rate already exist in a transmission queue) at every Tc, the traffic shaper checks to see if the transmission queue contains enough packets to send (that is, up to either Bc [or Bc plus Be] worth of traffic).

If the traffic shaper is not active (that is, there are no packets exceeding the configured traffic shaping rate in the transmission queue), the traffic shaper checks the number of tokens in the token bucket. One of the following occurs:

- If there are enough tokens in the token bucket, the packet is sent (transmitted).
- If there are not enough tokens in the token bucket, the packet is placed in a shaping queue for transmission at a later time.

Traffic Policing

In general, traffic policing allows you to control the maximum rate of traffic sent or received on an interface, and to partition a network into multiple priority levels or class of service (CoS).

Traffic policing manages the maximum rate of traffic through a token bucket algorithm. The token bucket algorithm can use the user-configured values to determine the maximum rate of traffic allowed on an interface at a given moment in time. The token bucket algorithm is affected by all traffic entering or leaving (depending on where the traffic policy with traffic policing is configured) and is useful in managing network bandwidth in cases in which several large packets are sent in the same traffic stream.

Traffic entering the interface with traffic policing configured is placed into one of these categories. Within these three categories, users can decide packet treatments. For instance, packets that conform can be configured to be sent, packets that exceed can be configured to be sent with a decreased priority, and packets that violate can be configured to be dropped.

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. In the most common traffic policing configurations, traffic that conforms is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs.

For Cisco IOS XR software, a single-rate, two-color policer is supported that provides one token bucket with two actions for each packet: a conform action and an exceed action. Traffic policing provides a certain amount of bandwidth management by allowing you to set the burst size (Bc) for the committed information rate (CIR). When the peak information rate (PIR) is supported, a second token bucket is enforced and the traffic policer is then called a two-rate policer.

Note The two-rate policer and two-token bucket algorithm is not supported on this release of Cisco IOS XR software.

Packet Marking Through the IP Precedence Value, IP DSCP Value, and the MPLS Experimental Value Setting

In addition to rate-limiting, traffic policing allows you to independently mark (or classify) the packet according to whether the packet conforms or violates a specified rate. Packet marking also allows you to partition your network into multiple priority levels or CoS.

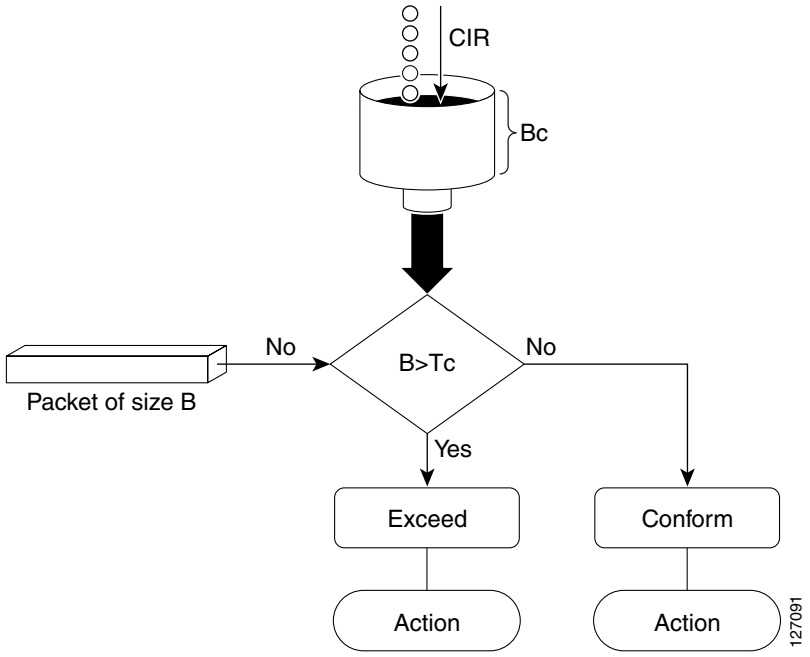
Use the traffic policer to set the IP precedence value, IP DSCP value, or Multiprotocol Label Switching (MPLS) experimental value for packets that enter the network. Then networking devices within your network can use this setting to determine how the traffic should be treated. For example, the Weighted Random Early Detection (WRED) feature uses the IP precedence value to determine the probability that a packet is dropped.

If you want to mark traffic but do not want to use traffic policing, see "Class-Based Packet Marking Feature and Benefits" to learn how to perform packet classification.

Traffic Policing Mechanism Regulates Traffic

Figure 3 illustrates how a single-rate token bucket policer marks packets as either conforming or exceeding a CIR.

Figure 3 How a Traffic Policing Mechanism Regulates Traffic



The time interval between token updates (Tc) to the token bucket is updated at the CIR value each time a packet arrives at the traffic policer. The Tc token bucket can contain up to the Bc value. If a packet of size B is greater than the Tc token bucket, then the packet exceeds the CIR value and a configured action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different configured action is performed.

Traffic Shaping Versus Traffic Policing

Although traffic shaping and traffic policing can be implemented together on the same network, there are distinct differences between them, as shown in Table 3.

Table 3 Differences Between Traffic Shaping and Traffic Policing

	Traffic Shaping	Traffic Policing
Triggering Event	• Occurs automatically at regular intervals (Tc). or Occurs whenever a packet arrives at an interface.	• Occurs whenever a packet arrives at an interface.
What it Does	• Classifies packets. • If a packet does not meet match criteria, no further action is taken. • Packets meeting match criteria are sent (if there are enough tokens in the token bucket) or Packets are placed in a queue for transmission later. • If the number of packets in the queue exceed the queue limit, the packets are dropped.	• Classifies packets. • If packet does not meet match criteria, no further action is taken. • Packets meeting match criteria and conforming to or exceeding a specified rate, receive the configured policing action (for example, drop, send, mark, then send). • Packets are not placed in a queue for transmission later.

How to Configure QoS Congestion Management on Cisco IOS XR Software

This section contains instructions for the following tasks:

- Configuring Guaranteed and Remaining Bandwidths, page QC-37 (required)
- Configuring Low-Latency Queueing with Strict Priority Queueing, page QC-40 (required)
- Configuring Traffic Shaping, page QC-42 (required)
- Configuring Traffic Policing, page QC-45 (required)

Configuring Guaranteed and Remaining Bandwidths

The **bandwidth** command allows you to specify the exact amount of bandwidth to be allocated for a specific class of traffic. MDRR is implemented as the scheduling algorithm.

The **bandwidth remaining** command specifies a weight for the class to the MDRR. The MDRR algorithm derives the weight for each class from the bandwidth remaining value allocated to the class. If you do not configure the **bandwidth remaining** command for any class, the leftover bandwidth is allocated equally to all classes.

Restrictions

The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.

A policy map can have all class bandwidths specified in kilobits per second or percentages but not a mixture of both in the same class.

The **bandwidth** command is supported only on policies configured on outgoing interfaces.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*kbps* | **percent** *value*}
5. **bandwidth remaining percent** *value*
6. **exit**
7. **class** *class-name*
8. **bandwidth** {*kbps* | **percent** *value*}
9. **bandwidth remaining percent** *value*
10. **exit**
11. **exit**
12. **interface** *type instance*
13. **service-policy** {**input** | **output**} *policy-map*
14. **end**
    or
    commit
15. **show policy-map interface** *type instance* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 4	**bandwidth** {*kbps* | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 50`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class1 is guaranteed 50 percent of the interface bandwidth.
Step 5	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth to various classes. • The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class2`	Specifies the name of a different class whose policy you want to create or change.
Step 8	**bandwidth** {*kbps* | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class2 is guaranteed 10 percent of the interface bandwidth.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`	Specifies how to allocate leftover bandwidth to various classes. • The remaining bandwidth of 40 percent is shared by class class1 (see Steps 4 and 5) and class2 in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 12	**interface** *type instance* **Example:** `RP/0/RP1/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode and configures an interface.

	Command or Action	Purpose
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • The traffic policy evaluates all traffic leaving that interface.

Step 14	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 15	**show policy-map interface** *type instance* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Low-Latency Queueing with Strict Priority Queueing

The priority command configures low-latency queueing (LLQ), providing strict priority queueing (PQ). Strict PQ allows delay-sensitive data, such as voice, to be dequeued and sent before packets in other queues are dequeued. When a class is marked as high priority using the **priority** command, we recommend that you configure a policer to limit the priority traffic. This configuration ensures that the priority traffic does not starve all of the other traffic on the line card, which protects low priority traffic from starvation. Use the **police** command to explicitly configure the policer.

Restrictions

Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is queued to the same single priority queue.

The **bandwidth**, **priority**, and **shape average** commands should not be configured together in the same class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police cir** *kbps* [**exceed-action** *action*]
5. **priority**
6. **exit**
7. **exit**
8. **interface** *type instance*
9. **service-policy** {**input** | **output**} *policy-map*
10. **end**
    or
    commit
11. **show policy-map interface** *type instance* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map voice`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class voice`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**police cir** *kbps* [**exceed-action** *action*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# police cir 250 exceed-action drop`	Configures traffic policing. • In this example, the low-latency queue is restricted to 250 kbps to protect low-priority traffic from starvation and to release bandwidth.
Step 5	**priority** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# priority`	Specifies priority to a class of traffic belonging to a policy map.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 8	**interface** *type instance* **Example:** `RP/0/RP1/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode, and configures an interface.
Step 9	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output voice`	Attaches a policy map to an input interface or an output interface to be used as the service policy for that interface. • The traffic policy evaluates all traffic leaving that interface.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show policy-map interface** *type instance* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Shaping

Traffic shaping allows you to control the traffic exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it.

Shaping performed on outgoing interfaces is done at the Layer 2 level and includes the Layer 2 header in the rate calculation.

Shaping performed on incoming interfaces is done at the Layer 3 level and does not include the Layer 2 header in the rate calculation.

Restrictions

The **bandwidth**, **priority**, and **shape average** commands should not be configured together in the same class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **shape average** {**percent** *value* | *kbps*}
5. **exit**
6. **exit**
7. **interface** *type instance*
8. **service-policy** {**input** | **output**} *policy-map*
9. **end**
   or
   commit
10. **show policy-map interface** *type instance* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**shape average** {**percent** *value* | *kbps*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# shape average percent 50`	Shapes traffic to the indicated bit rate according to average rate shaping in kilobits or the interface bandwidth in percentage.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 7	**interface** *type instance* **Example:** `RP/0/RP1/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • The traffic policy evaluates all traffic leaving that interface.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show policy-map interface** *type instance* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police** {**cir** {*kbps* | **percent** *percent*}} [**bc** *conform-burst*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action*] [**pir** *kbps*] [**violate-action** *action*]
5. **exit**
6. **exit**
7. **interface** *type instance*
8. **service-policy** {**input** | **output**} *policy-map*
9. **end**
   or
   commit
10. **show policy-map interface** *type instance* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 4	**police** {**cir** {*kbps* | **percent** *percent*}} [**bc** *conform-burst*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action*] [**pir** *kbps*] [**violate-action** *action*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# police cir 250 conform-action set mpls experimental topmost 3 exceed-action set mpls experimental topmost 4`	Configures traffic policing. The traffic policing feature works with a token bucket algorithm. The *action* argument is specified by one of the following keywords: • **drop**—Drops the packet. • **set dscp dscp-value**—Sets the DSCP value and sends the packet. • **set prec new-precedence**—Sets the IP precedence and sends the packet. • **set mpls experimental topmost**—Sets the EXP value on the MPLS packet topmost label • **set discard class**—Sets the discard class and QoS group identifiers on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets. **Note** The **pir** and **violate-action** keywords are not supported on the Cisco XR 12000 Series Router.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 7	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS 0/5/0/0`	Enters configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if) service-policy input policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • The traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show policy-map interface** *type instance* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuration Examples for Configuring QoS Congestion Management on Cisco IOS XR Software

This section provides the following configuration examples:

- Traffic Shaping for an Input Interface on Cisco IOS XR Software: Example, page QC-47
- Traffic Policing for a Bundled Interface on Cisco IOS XR Software: Example, page QC-48

Traffic Shaping for an Input Interface on Cisco IOS XR Software: Example

The following example shows how to configure a policy map on an input interface:

```
policy-map p2
 class voip
 shape average percent 20

!
interface bundle-pos 1
 service-policy input p2
 commit
```

```
RP/0/RP0/CPU0:Jun 8 16:55:11.819 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT : Configuration
committed by user 'cisco'. Use 'show configuration commit changes 1000006140' to view the
changes.
```

The following example shows the display output for the previous policy map configuration:

```
RP/0/RP0/CPU0:router# show policy-map interface bundle-pos 1 input

Bundle-POS1 input: p2
  Class voip
    Classification statistics                      (packets/bytes)      (rate - kbps)
            Matched                              : 0/0      0
            Transmitted                          : 0/0      0
            Total Dropped                        : 0/0      0
    Queueing statistics
            Vital           (packets)            : 0
    Queueing statistics
            Queue ID                             : 38
            High watermark  (packets)            : 0
            Inst-queue-len  (bytes)              : 0
            Avg-queue-len   (bytes)              : 0
            TailDrop Threshold(bytes)            : 47923200
            Taildropped(packets/bytes)           : 0/0
  Class default
    Classification statistics                      (packets/bytes)      (rate - kbps)
            Matched                              : 0/0      0
            Transmitted                          : 0/0      0
            Total Dropped                        : 0/0      0
    Queueing statistics
            Vital           (packets)            : 0
    Queueing statistics
            Queue ID                             : 36
            High watermark  (packets)            : 0
            Inst-queue-len  (bytes)              : 0
            Avg-queue-len   (bytes)              : 0
            TailDrop Threshold(bytes)            : 239616000
            Taildropped(packets/bytes)           : 0/0
```

Traffic Policing for a Bundled Interface on Cisco IOS XR Software: Example

The following example shows how to configure a policy map for a bundled interface. Note that for bundled interfaces, policing can be configured only in percentage and not kilobits per second:

```
policy-map p2
 class voip
 police cir 23425
!
interface bundle-poS 1
 service-policy input p2
 commit
RP/0/RP0/CPU0:Jun  8 16:51:36.623 : qos_ma[286]: %QOS-QOS_RC_QOSMGR-3-RC_BUNDLE_BW_NOPCT :
Absolute bw specified for bundle interfaces, use percentage values instead
RP/0/RP0/CPU0:Jun  8 16:51:36.624 : qos_ma[286]: %QOS-QOS-3-MSG_SEND_FAIL : Failed to send
message to feature rc while adding class. Error code - Invalid argument

% Failed to commit one or more configuration items during an atomic operation, no changes
have been made. Please use 'show configuration failed' to view the errors
!
```

An error occurred after the attempted commit of an invalid configuration.

```
!
!
```

```
policy-map p2
 class voip
 police cir percent 20
 commit
RP/0/RP0/CPU0:Jun  8 16:51:51.679 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT :
Configuration committed by user 'cisco'.   Use 'show configuration commit changes
1000006135' to view the changes.
 exit
exit
interface bundle-poS 1
 service-policy input p2
 commit
RP/0/RP0/CPU0:Jun  8 16:52:02.650 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT :
Configuration committed by user 'cisco'.   Use 'show configuration commit changes
1000006136' to view the changes.
```

The following example shows the display output for the successful policy map configuration in which policing was configured in percentage:

```
RP/0/RP0/CPU0:router# show policy-map interface bundle-poS 1

Bundle-POS1 input: p2
  Class voip
    Classification statistics                    (packets/bytes)      (rate - kbps)
            Matched                             : 0/0      0
    Policing statistics                          (packets/bytes)      (rate - kbps)
            Policed(conform)                    : 0/0   0
            Policed(exceed)                     : 0/0   0
            Policed(violate)                    : 0/0   0
            Policed and dropped                 : 0/0
  Class default
    Classification statistics                    (packets/bytes)      (rate - kbps)
            Matched                             : 0/0      0
            Transmitted                         : 0/0      0
            Total Dropped                       : 0/0      0
    Queueing statistics
            Vital           (packets)           : 0
    Queueing statistics
            Queue ID                            : 36
            High watermark  (packets)           : 0
            Inst-queue-len  (bytes)             : 0
            Avg-queue-len   (bytes)             : 0
            TailDrop Threshold(bytes)           : 239616000
            Taildropped(packets/bytes)          : 0/0
```

Where to Go Next

To configure WRED and tail drop, see the "Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software" module.

To configure traffic classification techniques, see the "Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software" module.

To read Cisco IOS XR QoS overview information, see the "Modular Quality of Service Overview on Cisco IOS XR Software" module.

Additional References

The following sections provide references related to implementing QoS congestion management on Cisco IOS XR software.

Related Documents

Related Topic	Document Title
Packet classification	*Configuring Modular QoS Packet Classification on Cisco IOS XR*, Release 3.2
WRED, RED, and tail drop	*Configuring QoS Congestion Avoidance on Cisco IOS XR Software,* Release 3.2
Cisco CRS-1 router getting started material	*Cisco CRS-1 Carrier Routing System Getting Started Guide,* Release 3.2
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS-XR Software* module of the *Cisco IOS-XR System Security Configuration Guide,* Release 3.2
Cisco IOS XR QoS overview information	*Modular Quality of Service Overview on Cisco IOS XR Software*, Release 3.2

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
There are no applicable MIBs for this module.	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport